METHODS AND APPARATUSES FOR MASTER/SLAVE DIRECT MEMORY ACCESS HARDWARE AND SOFTWARE CONTROL

The present invention relates generally to direct memory access (DMA) and, more particularly, to DMA hardware/software control techniques and structures that provide optimized data throughput.

Technologies associated with the communication of information have evolved
5   rapidly over the last several decades. One of these communication technologies involves the transfer of data between memory devices. For example, the memory systems of various computer processing systems typically include a primary memory device and one or more larger, less expensive secondary memory devices. This permits large quantities of information to be stored in secondary memory for loading into the primary memory device
10   when needed. Similar data transfers occur between memory devices in interconnected processing systems, which transfers are sometimes referred to as "master-slave" data transfers, and which are carried out over an interconnect device, e.g., a data bus. For example, the processor of a cellular telephone (master) may need to transfer data to and from subsystem processors, e.g., an antenna subsystem (slave).
15   One technique for performing such data transfers is known as direct memory access (DMA). DMA refers to a capability provided by some computer processing architectures that allows data to be sent directly from one memory device to another memory device without involving the main processor(s), thereby speeding up overall processing speeds. Usually a specified portion of memory is designated as an area to be used for DMA
20   transfers. One problem with master-slave DMA data transfers occurs when a large number of data blocks need to be transferred between the master and slave devices. When this arises, conventional DMA data transfer techniques are challenged to perform such transfers efficiently and typically confront various latency issues. One example of such a latency issue is shown in the timeline of Figure 1. Therein, a first DMA data block is transferred
25   from a master device to a slave device during the period DMA 1 and a second portion of the data block is transferred between the master and slave devices during the period DMA 2. Note that between time periods DMA 1 and DMA 2 there is a delay period denoted t1 in Figure 1. During this time, various handshaking activities occur between the master and slave devices. For example, at the end of DMA 1, the master device will signal a DMA
30   controller indicating that it has transferred the first portion of DMA data. The DMA controller will then reset register(s) used in the DMA data transfer process, e.g., an address

2

register. After the handshaking is completed, the second DMA transfer (DMA2) can commence. This conventional DMA technique suffers from, among other things, an inefficiency in DMA transfer time due to the delay period t1.

5   One partial solution to this problem involves using two memory portions for DMA transactions an example of which is described in U.S. Patent No. 6,341,318, the disclosure of which is incorporated here by reference. Therein, with reference to Figures 2(a) and 2(b), in iteration 0, memory bank A is loaded (LD0) with a first portion of data from a data block. In iteration 1, a next portion of data is loaded (LD1) into memory bank B while the previously loaded portion in memory bank A is processed (P0) and then stored (ST0). In

10   iteration 2, memory bank A is again loaded (LD2) with a third portion of data from a data block while the previously loaded portion in memory bank B is processed (P1) and then stored (ST1). The loading and processing/storing operations continue switching between each memory bank until all of the data block is loaded, processed, and stored. This technique may reduce latency associated with this type of DMA data transfer since the

15   processing of portions of oversized DMA data blocks can be performed in parallel. However, the technique described in the '318 patent does not address the afore-described problem of handling a large number of DMA transfers between master and slave devices. Under such conditions, even using two DMA memory locations may be insufficient to eliminate latency associated with setting up each transfer. Moreover, the '318 patent

20   provides relatively little control or status functionality in its register architecture. For example, no mechanisms are described for servicing higher level software protocols during the DMA transaction. As such, the DMA transaction architecture described in the '318 patent does not provide a DMA architecture that is sufficiently flexible to be reused without modifications between different hardware/software applications having different higher

25   level servicing needs.

       Accordingly, it would be desirable to provide more flexible techniques and devices for DMA data transfers which avoid the problems of conventional techniques.

       Systems and methods according to the present invention address this need and others by providing a robust DMA architecture including registers that flexibly support higher

30   level hardware/software servicing needs during a DMA transfer. Among other things, such registers include mailbox registers and scratch pad registers which provide DMA architectures and methods according to the present invention with the capability to provide multiple interrupt channels and overflow data transfer capabilities that permit disparate

3

hardware/software applications to reuse these DMA architectures with changes in their programming rather than changes in their architecture.

According to one exemplary embodiment of the present invention, a direct memory access (DMA) system for transferring information between a master device and a slave device comprises a first plurality of DMA channels for transferring information from the master device to the slave device; a second plurality of DMA channels for transferring information from the slave device to the master device; a first set of control registers which coordinate use of the first plurality of DMA channels to transfer information from the master device to the slave device; and a second set of control registers which coordinate use of the second plurality of DMA channels to transfer information from the slave device to the master device, wherein both the first set of control registers and the second set of control registers include at least one mailbox register for selectively providing an interrupt signal to a respective one of the master device and the slave device, in response to data written to the at least one mailbox register by the other of the master device and the slave device, during the DMA transaction.

According to another exemplary embodiment of the present invention, a method for direct memory access (DMA) information transfer between a master device and a slave device comprises the steps of: storing a first data portion received during a DMA transaction using a first DMA channel; storing a second data portion received during the DMA transaction using a second DMA channel; and selectively providing an interrupt signal to one of the master device and the slave device, by writing data in at least one mailbox register by the other of the master device and the slave device, during the DMA transaction.

The accompanying drawings illustrate exemplary embodiments of the present invention, wherein:

FIG. 1 depicts a timeline illustrating a DMA data transfer according to one conventional technique using a single memory area designated for DMA data transfers;

FIGS. 2(a) and 2(b) illustrate a conventional DMA data transfer using two memory areas;

FIG. 3 depicts a block diagram of a master-slave DMA architecture according to an exemplary embodiment of the present invention;

FIG. 4 shows an enlarged view of the slave portion of the DMA architecture of FIG. 3 including exemplary register groups;

4

FIG. 5 is a table providing an overview of various registers which can be included in DMA architectures according to exemplary embodiments of the present invention;

FIGS. 6(a)-6(l) illustrate exemplary individual register structures according to exemplary embodiments of the present invention;

FIG. 7 is a flowchart illustrating an exemplary method for transferring data from a master device to a slave device from the master's perspective according to an exemplary embodiment of the present invention;

FIG. 8 is a flowchart illustrating an exemplary method for transferring data from a master device to a slave device from the slave's perspective according to an exemplary embodiment of the present invention;

FIG. 9 is a flowchart illustrating an exemplary method for transferring data from a slave device to a master device from the slave's perspective according to an exemplary embodiment of the present invention; and

FIG. 10 is a flowchart illustrating an exemplary method for transferring data from a slave device to a master device from the master's perspective according to an exemplary embodiment of the present invention

The following detailed description of the invention refers to the accompanying drawings. The same reference numbers in different drawings identify the same or similar elements. Also, the following detailed description does not limit the invention. Instead, the scope of the invention is defined by the appended claims.

In order to provide some context for this discussion, an exemplary master-slave system in which techniques and devices according to exemplary embodiments of the present invention can be implemented is depicted in the block diagram of Figure 3. Therein, the master device 30 includes a processor 32, memory device 34 and master DMA controller 36 (master DMA controller 36 may be logically viewed as part of the hardware connect layer 38). Similarly, the slave device 40 includes its own processor 42, memory device 44 and slave DMA controller 46 (slave DMA controller 46 may also be logically viewed as part of the hardware connect layer 38). The hardware connect layer 38 can be implemented as, for example, any suitable data bus that can convey the depicted functionality, e.g., a master to slave interrupt signal, a master to slave DMA data transfer signal, a slave to master interrupt signal and a slave to master DMA data transfer signal. Devices 34 and 44 are not limited to implementation as memory devices *per* se and can, alternatively, be any other devices which use data transferred via a bus from another device.

Figure 4 illustrates the slave DMA controller 46 in more detail and, in particular, shows a general grouping of various registers which can be used in accordance with exemplary embodiments of the present invention to coordinate DMA data transfers. A first (optional) group 48 of registers includes hardware connection layer 38 specific registers

5   used by the master device 30 to control slave DMA and/or slave interrupt generation activity. For example, if an SDIO bus is used as hardware connect layer 38, the first group of registers 48 may include a register which indicates whether one bit mode or four bit mode is being used. Other examples include setting hardware specific registers to indicate whether DMA data transfers are occurring in full duplex mode or half duplex mode and/or

10   the operating frequency. Typically, the hardware specific registers 48 (if any) will be only programmed during an initialization phase. A second group of registers 50 include master to slave mailbox and scratch registers which are used by the master device 30 to transfer software specific information to the slave device 40. As mentioned earlier, mailbox and scratch registers provide, among other features flexibility for higher level software and

15   hardware protocols to operate during DMA activities. For example, when the master device 40 writes in a mailbox register, an interrupt signal can be sent to the slave processor 42. The generation of the interrupt signal responsive to writing in the mailbox register is programmable, as described below with respect to the control registers. The slave device 30 will then read the information stored in the mailbox register, which results in the interrupt

20   being cleared. The information written to mailbox registers according to exemplary embodiments of the present invention may vary widely depending upon the master and slave devices being interconnected by the DMA architectures. For example, if the master device is a cellular telephone's central processor and the slave device is an antenna subsystem, the master device could write instructions to the antenna subsystem into the

25   mailbox register(s), e.g., requesting the antenna subsystem to initiate low power consumption operation. Thus, the mailbox registers can be used to provide built-in interrupt channels which can be used in various ways depending upon the interconnected devices. The scratch registers are similar to the mailbox registers except that writing to a scratch register does not result in the generation of an interrupt signal. Thus, for example, the

30   scratch registers can be used to carry overflow data associated with a command transferred between the master and slave devices via the mailbox register. Alternatively, the slave device 40 can find master device status information in a scratch register.

6

Register group 52 depicts mailbox and scratch registers for the slave to master direction. This group of registers performs similar functions to those described above with respect to group 50. When the slave device 40 writes in one of these mailbox registers, an interrupt signal can be sent to master device 30, subject to control register programming.

5   When the master device 30 reads the corresponding mailbox, the interrupt will be cleared. Register group 54 depicts the DMA status/control registers which, among other things, control the amount of data that is transferred in one DMA cycle. In the master-to-slave direction, the control registers also indicate the address to which the data will be written, while in the slave-to-master direction the control registers also indicate the address from

10   which the data will be read.   As will be described in more detail below regarding individual registers and their functions, the status registers provide an indication of DMA activity as well as which interrupts are pending.

To better understand architectures according to exemplary embodiments of the present invention, a more detailed example of the various registers mentioned above will

15   now be provided with respect to Figures 5-6(l). Those skilled in the art will appreciate that the details associated with this example are purely illustrative in nature and can be varied. Among other things, the addresses and bit lengths are purely exemplary. Figure 5 provides a table listing the programmable registers on the slave side of the hardware connect layer 38 including their type. Each of the registers listed in the table of Figure 5 is depicted in more

20   detail in a corresponding one of Figures 6(a)-6(l).

Starting with Figure 6(a), an example of the master to slave interrupt status/control register is provided. This register type is also referred to herein as a "mailbox interrupt status/control register". This control/status register is used to control, and provide information regarding, the master-to-slave mailbox registers. Specifically, the slave device

25   40 can set a control bit associated with each of the master-to-slave mailbox registers to permit or deny interrupts to be generated when the corresponding mailbox register is written to by the master device 30. This feature can be used, for example, during initialization when disabling the mailbox interrupts may be desirable. After initialization, the slave device 40 may then enable the mailbox register interrupts. Status bits for each master-to-

30   slave mailbox are also provided in this register, which permit the slave device 40 to identify which mailbox or mailboxes caused it to receive an interrupt. Once read, these status bits are reset. Similarly, Figure 6(b) depicts an interrupt status/control register for the slave-to-master direction. Thus, the master device 30 can set a control bit associated with each of

the slave-to-master mailbox registers to permit or deny interrupts to be generated when the corresponding mailbox register is written to by the slave device 30. Status bits for each slave-to-master mailbox are also provided in this register, which permit the master device 30 to check to see if an interrupt is pending.

5   Figure 6(c) depicts an exemplary DMA status/control register. This register can include, among other things, bits to store version information, bits indicating DMA activity for each DMA memory location in each direction, and bits for controlling which of the two DMA memory locations will be used for a next DMA transaction. With respect to this latter function, it can be seen in Figure 6(c) that the values for bits 0 and 1 in this register 10   can be automatically inverted after each master-to-slave and slave-to-master transaction, respectively, to provide a round robin usage of the dual memory locations as described in more detail below. Figures 6(d)-6(g) provide examples of a master-to-slave mailbox register, a master-to-slave scratch register, a slave-to-master mailbox register and a slave-to-master scratch register respectively. Although multiple mailbox and scratch registers can be 15   provided in each direction for DMA architectures according to the present invention, only one of each type/direction is described here since each of these types of registers can have the same structure and functionality. The number of bits allocated for storage of mailbox code and/or scratch values can be more or fewer than the 10 bits indicated in the examples shown in Figures 6(d)-6(g).

20   Figure 6(h) illustrates an exemplary DMA interrupt status/control register. This register provides for programmable enablement/disablement of the interrupt signals which are used to signal the completion of a DMA transfer. One bit is provided for each DMA memory location for each direction, for a total of four control bits in this exemplary embodiment of the present invention. Similarly, four bits provide status information for 25   each DMA memory location for each direction regarding whether or not an interrupt has been set. The DMA interrupt status/control register can also be used to provide other functionality, for example providing an indicator regarding whether 16 bit data values are transmitted with their most significant bit (MSB) first or least significant bit (LSB) first. Figures 6(i)-6(l) illustrate exemplary master-to-slave DMA address register, master-to-slave 30   DMA size register, slave-to-master DMA address register and slave-to-master DMA size register, respectively. One address and one size register can be provided for each DMA memory location used to transfer DMA data to specify the address and size of each memory

8

location. According to this exemplary embodiment, therefore, eight of these registers can be used as seen in Figure 5.

Having described a register architecture according to exemplary embodiments of the present invention, exemplary methods for transferring data using such an architecture will now be discussed with reference to the flowcharts of Figures 7-10, wherein Figures 7 and 8 depict a master-to-slave DMA transfer method and Figures 9 and 10 depict a slave-to-master DMA transfer method. On the master device's side, at step 70, the master device 30 first waits for the slave device 40 to initialize. This can be accomplished by, for example, the master device 30 polling one of the slave-to-master scratch registers and waiting for the slave device 40 to have written a predetermined value in that register. Next, at step 72, the master device 30 initializes the interrupts related to the master-to-slave mailboxes (Fig 6a). The slave device 40 uses these interrupts to indicate that a master-to-slave DMA channel has been initialized and is ready to receive data. The master device 30 then waits for data to send to the slave device 30 at step 74. At this step, the master device 30 knows that the slave device 40 has two master-to-slave DMA channels available. When master device 30 has data that needs to be transferred to the slave device 40, it then performs a DMA transfer at step 76. In a typical implementation of the slave DMA controller 46, the master device 30 does not have to keep track of which DMA channel the DMA data needs to be sent. The slave DMA controller 46 will automatically direct the incoming master-to-slave DMA data to the appropriate, next DMA channel. For this purpose the slave DMA controller 46 can have some hardware connect layer specific registers which are accessible by the master device 30 to enable this functionality. At step 78, the master device 30 is waiting for more data to send to the slave device 30 or for reception of an interrupt indicating that the slave device 30 has freed up the previously used DMA channel. Depending on which of the two indications comes in first, the master device 30 will change to the appropriate state, i.e., return to step 74 wherein two DMA channels are available or move on to step 80. Typically a slave-to-master mailbox interrupt is used by the slave device 40 to indicate to the master device 30 that a channel has been freed up again. However, there are other mechanisms available as well. For example, the master device 30 can periodically poll a scratch register, where the slave programmed value can be used to derive if the slave device 40 has any master-to-slave DMA channels available. Assuming that the flow next moves to step 80, the master device 30 once again has data that needs to be transferred to the slave device 40, and it sends the data via the DMA channel selected by the slave DMA controller 46.

9

The flow then proceeds to step 82, wherein the master device 30 knows that the slave device has no free master-to-slave DMA channels available. The master device 30 then waits for an indication (interrupt or polling) from the slave that a DMA channel is freed up again and the flow returns to step 78.

5   Figure 8 shows an exemplary process which occurs during a master-to-slave DMA transfer from the slave device 40's perspective. Therein, the slave device 40 initializes the master-to-slave interface at step 84. This step might, for example, include programming some hardware connection layer specific registers 48. The generation of master-to-slave DMA interrupts (Fig. 6(h)) will also typically be enabled at step 84, as well as setting the

10   locations for the two master-to-slave DMA channels in the respective registers. When all of the initialization is done, the slave device 40 indicates to the master device 30 that the initialization has finished by, for example, writing a predetermined value to one of the scratch registers. Next, at step 86, the slave device 30 waits for the master device's DMA data. In this state the slave has two master-to-slave DMA channels available. When the

15   slave device 40 receives a master-to-slave DMA end interrupt the slave device 40 can determine from the DMA status register (Fig. 6(h)) for which master-to-slave DMA channel the interrupt was generated. With that information the slave can find the memory location of the DMA data and the size from the channel corresponding size and address registers (Figs. 6(i) and 6(j)). The slave software, e.g., antenna subsystem software, can then be

20   triggered to process the received data.

          The process then moves to step 88. Therein, the slave device 40 can check to determine if it has a new memory location available to reprogram the memory address (Fig. 6(i))) for the DMA channel which has previously received DMA data, e.g., if that DMA channel has not been cleared yet. If so, the address register for this DMA channel is

25   updated, and a signal is sent to the master device 30 to indicate that a new channel is available. This can be accomplished by, for example, generating a slave-to-master mailbox interrupt to indicate to the master that a DMA channel has been freed up again. As mentioned above, however, polling or hybrid polling/interrupt based schemes can be used for this purpose as well. If no memory location was available and a new DMA end interrupt

30   is received, the slave device 30 can derive from the DMA status register (Fig. 6(h)) for which master-to-slave DMA channel the interrupt was generated. With that information the slave device 30 can find the memory location of the DMA data and the size from the

corresponding size and address registers (Figs. 6(i) and 6(j)). The slave software that will process the just received data will then be triggered.

The slave device 30 then moves to step 90 wherein it again waits for new memory to become available in which master-to-slave DMA package could be stored. When this memory becomes available, the address register for the next master to slave DMA channel is updated, and a trigger is generated to the master to indicate that a new channel is available. Those skilled in the art will appreciate that, instead of only a round robin approach using two memory locations for DMA transfers, the present invention can use three or more memory locations to implement the two DMA channels described above. For example, at steps 88 and 90 memory addresses which are different than those used at step 84 can be programmed into one or both of the address registers. Using more than two memory locations for master-slave DMA transactions can further reduce the latency, particularly for DMA transactions involving numerous data packages.

Figure 9 depicts a slave-to-master DMA data transfer method according to an exemplary embodiment of the present invention from the slave device 40's point of view. Therein, at step 92, the slave device 30 initializes the master-to-slave interface. This step can, for example, include programming some hardware connection layer specific registers 48. The master-to-slave DMA interrupts (Fig. 6(h)) can be enabled depending on later described communication algorithms. When the initialization is completed, the slave device 40 needs to indicate this to the master device 30. This can be accomplished by writing a predetermined value to one of the scratch registers. At step 94, the slave device 40 is waiting for an indication that the master device 40 has been initialized. This can be accomplished by, for example, the master device 30 polling one of the master-to-slave scratch registers and waiting for the slave to write a predetermined value in that register.

Next, at step 96, the slave device 40 waits for data to send to the master device 30 and has two slave-to-master DMA channels available. When that data becomes available, at step 98, the slave device 40 initializes the address and size registers (Figs. 6(k), 6(l)) for the next DMA channel (Fig. 6(c)), and generates an indication to the master that a next slave-to-master DMA transfer has been setup. This indication can be an interrupt that is generated by writing to a mailbox register or updating a value in a scratch registers that the master polls, from which the master device can derive that new DMA data transfer has been setup. Upon reception of this indication, the master will initiate the master-to-slave DMA transfer. As mentioned earlier, the slave DMA controller 46 can automatically retrieve the slave-to-

master DMA data from the valid channel without any intervention from the master device 30.

At step 100, the slave device 40 is waiting for more data to send to the master or for the reception of an indication that the master device 30 has retrieved the slave-to-master

5 DMA data of the initialized channel. This indication can be provided by using the TX DMA end interrupt in the register of Fig. 6(h). This interrupt will indicate which channel data has been retrieved. Another method for providing this indication is that after retrieving the DMA data, the master device 30 writes to a master-to-slave mailbox register, which will generate an interrupt for the slave device 40. A third method is that the slave device 40 polls

10 a master-to-slave scratch register, whose value will be updated by the master device 30 after retrieving the DMA data. Assuming that the slave has more data for the master prior to receiving an indication that the previous slave-to-master DMA transfer has been completed, the process transitions to step 102. Therein, the slave device 40 initializes the address and size registers (Figs. 6(k) and 6(l)) for the still available DMA channel, and generates a

15 previously described indication to the master device 30 that a next slave-to-master DMA transfer has been setup. The flow then moves to step 104, wherein the slave device 40 has no free slave-to-master DMA channels available and, therefore, waits for an indication (interrupt or polling) from the master device 30 that a DMA channel is freed up again.

Turning now to the master device's perspective of the slave-to-master DMA transfer

20 in Figure 9, the master device 30 waits for the slave device 40 to initialize at step 106. This can be accomplished by, for example, the master device 30 polling one of the slave-to-master scratch registers and waiting for the slave device 40 to write a predetermined value in that register. Next, the master device 30 initializes the slave DMA controller 46 at step 108. This step can, for example, include programming hardware connection layer specific

25 registers 48. The generation of slave-to-master DMA interrupts (Fig. 6(h)) will also be enabled (if needed). When the initialization is completed, the master device 30 indicates this to the slave device 40. This indication can, for example, be accomplished by writing a predetermined value to one of the master-to-slave scratch registers.

The master device 30 then waits for a slave-to-master DMA data indication at step

30 110. In this state there are no slave-to-master channels pending with data to be read. There are several ways that the data indication can be generated. For example, the slave device 40 can indicate that it has data ready to transfer via an interrupt that is generated by the slave writing to a slave-to-master mailbox register. Alternatively, the slave device 40 can update

12

a value in a scratch register that the master device 30 polls, from which the master can determine that new DMA data has been setup. Upon reception of this indication, the master device 30 will initiate the master-to-slave DMA transfer. As mentioned earlier, according to exemplary embodiments of the present invention, the master device 30 does not have to

5 keep track of the DMA channel from which the DMA data needs to be retrieved. Instead, the slave DMA controller 46 can automatically retrieve the slave-to-master DMA from the valid channel. At step 112, the slave-to-master DMA transfer is ongoing. The master device 30 is waiting for either the end of DMA transfer indication (which is generated internally in the master), or a new indication from the slave device 40 that yet another DMA

10 channel has been setup. When the end of DMA transfer indication is received, the master device 30 can trigger the function that will process the retrieved data. Depending on which 'end of slave to master DMA transfer' indication the slave uses, the master can write to a master-to-slave mailbox or scratch register to indicate that this transfer finished. When a new slave-to-master DMA data indication is received by the master device 30, and the

15 previous DMA transfer has not yet finished, the process moves to step 114. Therein, the slave-to-master DMA is ongoing. The master is waiting for the end of DMA transfer indication (which is generated internally in the master). When the end of DMA transfer indication is received, the master can trigger the function that will process the retrieved data. Again, depending on which 'end of slave to master DMA transfer' indication the

20 slave uses, the master device 30 can write to a master-to-slave mailbox or scratch register to indicate that this transfer finished, which returns the flow to step 100.

        In the exemplary DMA methods described above, it is assumed that the slave DMA controller 46 has four parallel DMA channels, but the master device 30 does not. If the master device 30 also has four parallel DMA channels, the master device's flow diagrams

25 would be similar to those of the slave device 40. Additionally, providing four parallel DMA channels for the master device would enable the master device 30 to setup several DMA transfers in parallel. Although the foregoing examples depict two DMA channels in each direction, those skilled in the art will appreciate that DMA architectures according to the present invention are not so limited. Instead, embodiments of the present invention

30 could contain four, eight, sixteen or more DMA channels in each direction with a corresponding increase in the number of registers used to implement the architecture.

        According to exemplary embodiments, the master device 30 initiates slave-to-master DMA transfers. If the slave device 40 has not initialized any slave-to-master DMA

channels with a new memory address and size, the slave DMA controller 46 will automatically respond to the slave-to-master DMA transfer request with the indication that the DMA size is zero. As mentioned above, initiating a slave-to-master DMA data transfer can be accomplished in at least three different ways. First, a polling based initialization can
5   be used. If this technique is used, then the master device 30 will periodically start a slave-to-master DMA command sequence and will read, among other things, the slave-to-master size registers to determine if any DMA data has been stored for transfer. If data is available, then the process outlined above is performed, otherwise the master device 30 awaits the next polling interval.

10   Alternatively, an interrupt based initialization can be used. After the slave device 40 has programmed the slave-to-master address and size registers, a mailbox register is then written to in order to generate an interrupt to the master device 30. This provides yet another example of the flexibility created by providing mailbox registers to DMA architectures according to the present invention. On reception of the interrupt generated by
15   the mailbox register, the master will initiate a slave-to-master DMA sequence. Different mailbox registers can be used to initiate transfers from two different DMA memory locations. The slave device 40 should first check the interrupt status bits in the mailbox interrupt status/control register to ensure that an interrupt is not already pending for the mailbox registers it intends to use to signal the master device 30 to initiate the slave-to-
20   master DMA data transfer.

        Thirdly, a hybrid polling/interrupt based initialization technique can be used for slave-to-master DMA transfers according to exemplary embodiments of the present invention. To begin, the slave device 40 again writes to a mailbox register to generate an interrupt to the master 30 which is interpreted as a request to initiate a slave-to-master DMA
25   transfer. While the transfer is occurring from the first memory location, the slave device 40 programs the registers and second memory location with the next DMA package to be transferred. However, instead of using another mailbox register to send an interrupt signal to the master device 30 to start the transfer of the second data portion, the master device 30 automatically initiates the second slave-to-master transfer sequence. The slave device 30
30   then uses the interrupt it receives upon completion of the first transfer to begin preparing the third data portion for transfer. This process continues until such time as the slave device 40 runs out of data associated with this DMA transaction at which time it sets the size register

to zero. When the master device 30 requests the next block of data it will receive the size

zero indication and end the DMA transaction.

5   Those skilled in the art will appreciate that the foregoing DMA architectures and

methods provide sufficient programmability for interrupts and data transfers that they lend

themselves to reuse without architectural changes between different types of master devices,

slave devices, hardware interconnects and software applications. The ability to program

the mailbox registers to selectively enable or disable interrupt generation, trigger the

reading of scratch registers, etc., allows software developers to reuse code associated with

this DMA architecture since it can maintain its register structure across different

10 implementations.

      The above-described exemplary embodiments are intended to be illustrative in all

respects, rather than restrictive, of the present invention. Thus the present invention is

capable of many variations in detailed implementation that can be derived from the

description contained herein by a person skilled in the art.   All such variations and

15 modifications are considered to be within the scope and spirit of the present invention as

defined by the following claims. No element, act, or instruction used in the description of

the present application should be construed as critical or essential to the invention unless

explicitly described as such. Also, as used herein, the article "a" is intended to include one

or more items.